Eclipse Plug-in Development Research Project

Report 5

UNIVERSITY AT BUFFALO

July 17, 2015
Authored by: Chern Yee Chua

Eclipse Plug-in Development Research Project

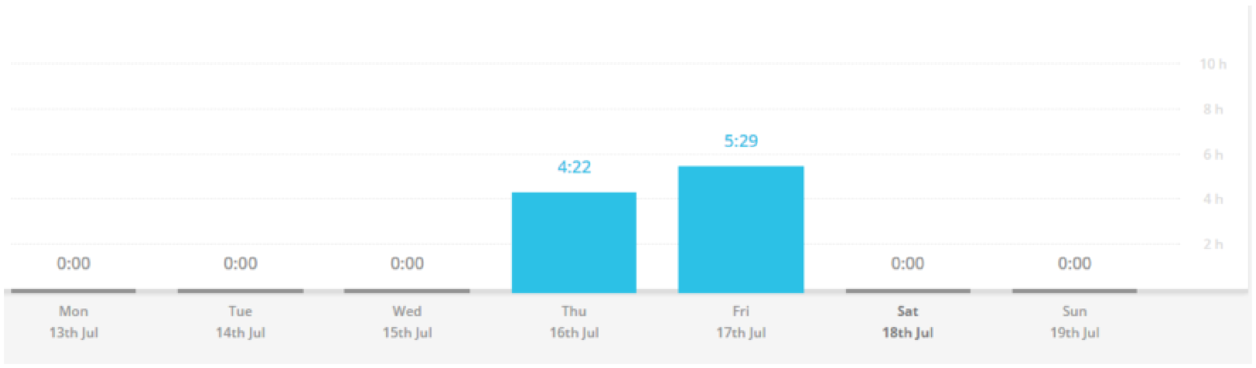
Report 5

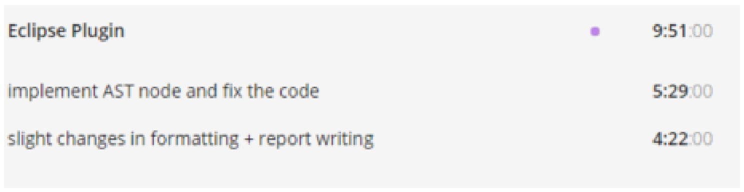
Introduction

In this week, I have changed the output file method as well as some tiny output data format.

Synopsis

I spend roughly 10 hours for this week.



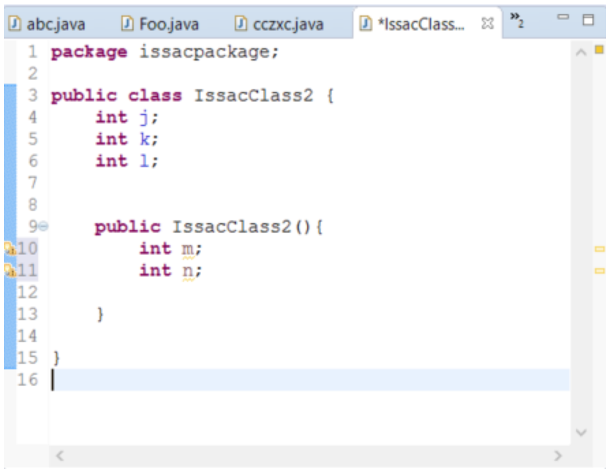


Eclipse Plugin	●	9:51:00
implement AST node and fix the code		5:29:00
slight changes in formatting + report writing		4:22:00

Contents

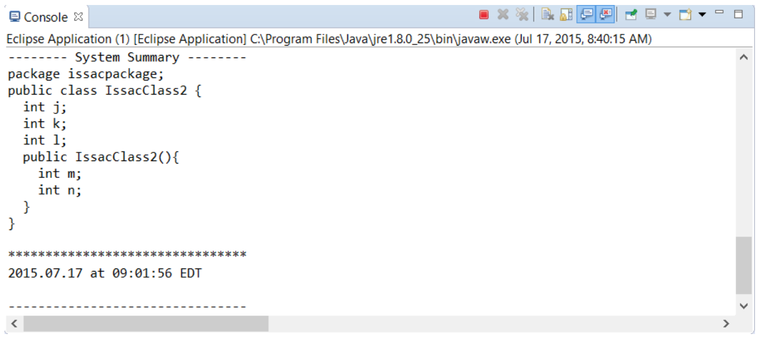
As Professor Alphonce suggested, I extend the system output file to the previous output file (as opposed to override the file for every update). The output file should be continuous.

Besides that, I use a "cheating way" to display all the elements in within CompilationUnit as demonstrated below:



```java
package issacpackage;

public class IssacClass2 {
    int j;
    int k;
    int l;


    public IssacClass2(){
        int m;
        int n;

    }

}
```

Snapshot above shows the Java class file. Below shows the output system console:

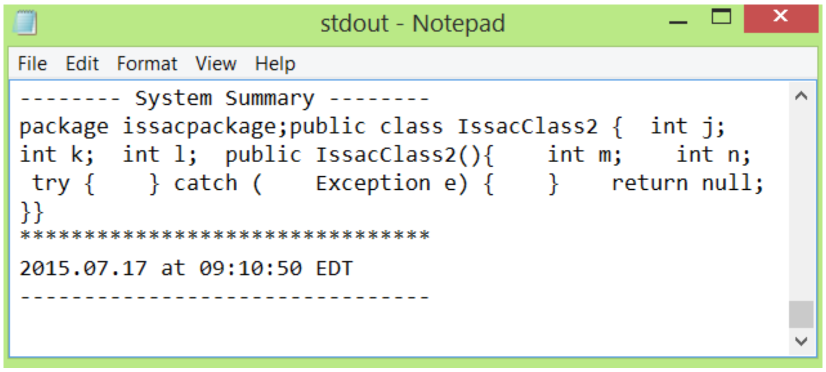


```
Eclipse Application (1) [Eclipse Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (Jul 17, 2015, 8:40:15 AM)
-------- System Summary --------
package issacpackage;
public class IssacClass2 {
  int j;
  int k;
  int l;
  public IssacClass2(){
    int m;
    int n;
  }
}

********************************
2015.07.17 at 09:01:56 EDT

--------------------------------
```

As you can see, it outputs pretty much the same as what is written on the editor. This hack is accomplished by printing out the CompilationUnit node (inspired by MethodDeclaration node). In this way, we are not only able to record the elements inside the method, we can also record basically everything inside this Java file (except for comments). Since this is pretty much the same as traversing all the JavaElementDelta children, I think this is a better way to display all necessary information. And as before, this system summary will be printed out for a certain amount of time when user make changes. Extended from previous work, when user add or remove elements, the basic elements such as class, method, variable, and etc. can be detected when changes are made. For those which can't be detected such as the variable inside the method, we can rely on the periodical system summary, at least for now.

Questions

Do you think this is a good approach? I mean, I can cast the element node to String and from there I can simply dissect and extract each component and store them in array individually.



```
-------- System Summary --------
package issacpackage;public class IssacClass2 {  int j;
int k;  int l;  public IssacClass2(){    int m;    int n;
 try {    } catch (    Exception e) {    }    return null;
}}
********************************
2015.07.17 at 09:10:50 EDT
--------------------------------
```

For example, the snapshot above (taken from output file) shows the system summary. I can tokenize them since it is just a bunch of string, and then use HashMap to map them to different array list. For example, an int array to keep track of how many integer variable / return type signature, method array, and etc.

What do you think? Necessary or redundant?

Conclusion

I am sorry for not doing much for the project as I was away from Buffalo for a week. I have reached the bottleneck that I don't know what I can do to make it better. For the case above, printing out the system summary is a great way to see someone's code from time to time, but it is not great to keep track of the elements since it's just printed out whatever it is on the editor. Or if I manage to make this plugin sorting all the detected elements and categorizing them nicely just like ASTView, we will still have a hard time to see how student works on their code. The objective of the project is a bit unclear on what we really want to see. So far it is recording as much information as it can, which is good, but what are we going to do with all these data? Do we just see how student writes code from time to time or do we let the computer to do checking and comparing things like that? This is my doubt for now. And of course, I still have to make it installed on eclipse so it runs when eclipse starts.

Thank you for your time!